Fair Social Choice in Dynamic Settings

Rupert Freeman, Seyed Majid Zahedi, and Vincent Conitzer

Abstract

We study a dynamic social choice problem in which an alternative is chosen at each round according to the reported valuations of a set of agents. In the interests of obtaining a solution that is both efficient and fair, we aim to maximize the *Nash social welfare*, which is the product of all agents' utilities. We present three novel rules and discuss some of their properties. Two are greedy algorithms and the third attempts to explicitly learn the distribution over inputs, updating its decisions by solving a convex program at each round. We also take a more generally applicable algorithm from existing literature and apply it to our problem. Finally, we compare all four algorithms against the offline optimal solution in simulations.

1 Introduction

Fairness is a topic of rapidly increasing interest in social choice. On the one hand, there has been much recent interest in the fair allocation of resources—cake cutting [28] as well as other models [16, 25]. On the other hand, in voting, fairness considerations have received attention in selecting a committee of candidates, in the form of a focus on the voters being *represented* in the committee [12, 23, 8].

A classical approach to obtaining a fair outcome in a context where agents have utility functions is to maximize the *Nash social welfare* [24], which is the product of the agents' utilities. One attractive feature of using the Nash social welfare is scale invariance: if an agent doubles all her utilities (or, equivalently, changes the units in which she expresses her utilities), this does not change which outcomes maximize the objective (the Nash social welfare is not however stable under additive transformations, where an agent simply adds some constant value to all her reported utilities).

In life, it is often difficult to make a completely fair decision in a single-shot context; often, every option will leave some agents unhappy. Fortunately, we can often address this over time—we will go to my most preferred restaurant today, and to yours next week. Achieving fairness over time is the topic of our paper. Ours is certainly not the first work to consider fairness or social choice in dynamic settings; see, for example, [27, 19, 6].

When we make multiple decisions over time, we could simply maximize the Nash welfare in each round separately. But it is easy to see that this can lead to dominated outcomes. For example, suppose there are two agents, and we can choose an alternative that gives one a reward of 3, and the other a reward of 0; or vice versa; or an alternative that gives each of them 1. Within a round, the last alternative maximizes Nash welfare; but if this scenario is repeated every round, then it would be better to alternate between the first two alternatives, so that each agent obtains 1.5 per round on average. Of course, *initially*, say in the first round, we may not realize we will have these options every round, and so we may choose the last alternative; but if we do have these options every round, we should *eventually* catch on to this pattern and start alternating. Ideally, we would maximize the long-term Nash welfare, that is, the product of the long-run utilities (which are the sums of each agent's rewards), rather than, for example, the sum of the products within the rounds. Of course, if there is uncertainty about the options that we will have in future rounds, we cannot expect to get the same Nash welfare that we could obtain with perfect foresight. For example, we may choose to make an agent happy this round, and then later realize that in typical rounds, this agent is very easy to make happy and we should have focused our efforts on an agent that is more difficult to make happy. While such scenarios are inevitable, we do want to adapt and learn over time and thereby approximate the ideal Nash welfare.

In this work, we do not focus primarily on strategic concerns . Of course it is fairly common to ignore strategic concerns in the social choice literature, but we do think this is an important topic for future work (and we discuss some directions in Section 8). On the other hand, there are also important contexts where strategic concerns do not come into play. For example, instead of considering a setting where there are multiple *agents* that have different utility functions, we can consider a setting where there are multiple *objectives* that each alternative contributes towards. For example, consider faculty hiring. Suppose the three objectives that we want our faculty hires to contribute to are research, teaching, and service; moreover, suppose that at the time of hiring we can predict well how much each candidate would contribute to each of these objectives, if hired. Then, it stands to reason that, one year, we may hire a top researcher that we do not expect to contribute much to our teaching or service objectives. But we would be loath to make such a decision *every* year; having hired a few top researchers who are not good at teaching or service, pressure will mount to address these needs. This fits well into our framework, if we simply treat each of the three objectives as an agent that is "happy" with an alternative to the extent to which it addresses the corresponding objective. In particular, note that the fact that objectives are measured in incomparable units – for example, we might measure research crudely by number of top-tier publications, and teaching crudely by course evalation scores – poses no problem to our methodology, since this methodology can anyway address agents measuring their utilities in different units. (Since we are not in a setting with a numeraire, there is no reason their utilities should have similar units.) Thus, a reader who insists on game-theoretic modeling in the case of agents with utility functions may instead substitute this modified interpretation of addressing multiple objectives everywhere in our paper.

The rest of the paper is organized as follows. In Section 2 we introduce notation and preliminaries. In Section 3 we present two simple greedy algorithms for choosing alternatives, and provide intuitive interpretations of them. We make a computational distinction between them and provide an axiomatic justification for one of them. In Section 4 we present an algorithm which can be seen as an approximation to the optimal solution when T is infinite. In Section 5 we present an existing algorithm with good regret guarantees for a more general class of stochastic optimization problems.

After presenting the algorithms, we evaluate them on simulated data in Section 6. Finally, in Section 7 we discuss specific applications of our methodology, including to voting, and conclude in Section 8.

Related work: Parkes and Procaccia [27] examine a similar problem by modeling agents' evolving preferences with Markov Decision Processes, with a reward function defined over states and actions (alternatives). However, their goal is to maximize the sum of (discounted) rewards and they do not explicitly consider fairness as an objective. Kash, Procaccia and Shah [19] examine a model of dynamic fair division where agents arrive at different points in time and must be allocated resources; however, they do not allow for the preferences of agents to change over time as we do. A recent paper by Aleksandrov et al. [6] considers an online fair division problem in a setting where items appear one at a time, and agents declare yes/no preferences over that item. In our setting, each round has many alternatives and we allow agents to express more general utilities. Our work is related to the literature on dynamic mechanism design (see, e.g., [26] for an overview), except that we do not consider monetary transfers. Guo, Conitzer and Reeves [17] consider a setting similar to ours, also without money, except that they are not explicitly interested in fairness, only welfare, and focus on incentive compatibility.

2 Preliminaries

Consider a set of n agents and let A be a set of m possible alternatives.[1] At every round $t = 1, \dots, T$, agents report their valuation for every alternative. In this paper we allow the valuations to be integers in the range 0 to K for some finite K (therefore we can achieve arbitrarily fine granularity by

[1]For simplicity of presentation, we define the set of alternatives to be static. However, all of our algorithms and results hold if the set of alternatives, and even the number of alternatives, changes from round to round.

allowing K to be large). Thus the input at every round is a matrix $V_t \in \mathbb{Z}^{n\times m}_{\geq 0, \leq K}$. For every round t, a *Dynamic Social Choice Function (DSCF)* chooses a single alternative, corresponding to a column of V_t, which we denote by v_t. Importantly, the problem is *online*, so we may only use information up to time t in order to choose v_t.

The valuation of agent i for the alternative j at time t is $V_t(i, j)$, and at each round we can think of an agent's valuation vector, $V_t(i, \cdot)$, as their reported valuation for each alternative. Although the columns of V_t are formally indexed by alternatives, we will often refer to the $\mathrm{vec}u_t^{\mathrm{avg}}tor\mathrm{V}(\cdot, j)$ simply as j when there is no risk of confusion. Thus the valuation of agent i for alternative v_t will be denoted by $v_t(i)$. We define a vector of *accrued rewards at round* t, u_t, where the accrued reward of agent i at round t is the sum of agent i's valuations for the chosen alternatives up to and including round t, $u_t(i) = \sum_{t'=1}^{t} v_{t'}(i)$. We will most often be interested in an agent's accrued reward *before* the start of round t, $u_{t-1}(i)$. The average utility of the agents over the first t rounds is given by $u_t^{\mathrm{avg}} = \frac{1}{t}u_t$.

A DSCF is *anonymous* if applying permutation σ to the rows of V_t, for all t, does not change the chosen alternative v_t, for any t. A DSCF is *neutral* if applying permutation σ to the columns of V_t, for all t, results in choosing alternative $\sigma(v_t)$ for all t. For the rest of this paper we only consider anonymous, neutral DSCFs. The DSCFs that we discuss are presented in a way that naturally allows ties between alternatives. We think of the mechanisms choosing a set of possible alternatives, and then choosing a single alternative from the set arbitrarily.

The *Nash social welfare (NSW)* of utility vector v, $NSW(v)$, is defined to be the product of the agents' utilities, $NSW(v) = \prod_{i=1}^{n} v(i)$. The NSW is frequently used as an objective in the fair division literature as it strikes a balance between maximizing efficiency and fairness (for recent examples in the computer science literature, see [13, 14, 29]). One further nice property of NSW is that it is *scale-free*, meaning that the optimal choice of alternative is unchanged if some agent(s) report valuations on different scales from others. Our aim is to maximize the NSW of the average utility across all T rounds, $NSW(v_T^{ave})$. Some of our algorithms involve the use of convex programming, which requires a concave objective function to maximize. Unfortunately, NSW is not a concave function, however $\ln(NSW)$ is. Thus, we will interchangeably talk about maximizing $\ln(NSW(u_t^{\mathrm{avg}})) = \ln\left(\prod_{i=1}^{n} u_t^{\mathrm{avg}}(i)\right) = \sum_{i=1}^{n} \ln(u_t^{\mathrm{avg}}(i))$. Since ln is an increasing function, the solution maximizing $\ln NSW(v)$ is the same as the solution maximizing $NSW(v)$.

The benchmark algorithm is the opfimal offline algorithm, where an offline instance of the problem is given by the set of matrices $\{V_t\}_{t\in\{1,\ldots,T\}}$. The offline problem can be solved via the following mixed integer convex program:

$$\text{Maximize} \sum_{i=1}^{n} \ln\left(\sum_{t=1}^{T}\sum_{j=1}^{m} x_{tj} V_t(i,j)\right) \tag{1}$$
$$\text{subject to} \sum_{j=1}^{m} x_{tj} = 1 \quad \forall t, \qquad x_{tj} \in \{0,1\} \quad \forall t, j$$

where x_{tj} is a binary variable denoting whether or not alternative j is chosen at time t. The constraint simply says that for each t, we must choose exactly one alternative. We denote the optimal Nash social welfare by OPT (thus the optimal objective value achieved by convex program 1 is $\ln(\text{OPT})$).

The details of all missing proofs can be found in the appendix.

3 Greedy Algorithms

In this section we present two simple greedy algorithms. We note that, although these algorithms are designed to give an approximate solution to that which maximizes Nash welfare, much of this section is devoted to showing that the algorithms satisfy desirable properties as algorithms in their own right. Such an approach is not new in computational social choice – for other papers that treat approximation algorithms as distinct voting rules see, for example, [10, 11, 15]. The first algorithm, GREEDY, simply chooses v_t to maximize $NSW(u_t^{\text{avg}})$. The only subtlety is that in the early rounds it may not be possible to give all agents non-zero utility. Therefore it may be the case that $NSW(u_t^{\text{avg}}) = 0$ for all choices of v_t, even when one allocation is clearly better than all others. We address this by randomly allocating some small 'hallucinated' utility, ϵ_i, to those agents with zero accrued reward at each round (not necessarily the same to each agent). GREEDY is presented as Algorithm 1. The bounds on ϵ_i in Line 4 are used to ensure that no agent with zero accrued reward is given too high priority over any other agent with zero accrued reward, and will be helpful in proving Proposition 1 and Theorem 1. Let $0 < x < \frac{1}{2^n(K+1)^{n(n+1)}}$.

Algorithm 1 GREEDY

```
1: Initialize $u_0 = (0, \dots, 0)$
2: for $t = 1, \dots, T$ do
3:     for $i = 1, \dots, n$ do
4:         Set $x \leq \epsilon_i \leq 2(K+1)^{n-1}x$
5:     end for
6:     Choose $v_t \in \text{argmax}_{v \in V_t} \prod_{i=1}^{n} (\max\{u_{t-1}(i) + v(i), \epsilon_i\})$
7:     $u_t = u_{t-1} + v_t$
8: end for
```

The next algorithm, LINEARGREEDY, is a linear version of GREEDY which assigns each agent a weight w_i equal to the inverse of their accrued utility at the start of each round and simply chooses $v_t = \text{argmax}_{v \in V_t} w \cdot v$. This algorithm is known as the Proportional Fair algorithm in the networking community (see, among many others, [32, 18]) as, at each round, it maximizes the sum of the percentage increases in accrued reward across all agents.

Again, it is necessary to hallucinate small utility δ_i for all agents with zero accrued reward. Here, the upper bound on δ_i is chosen so as to guarantee that at least one agent with zero accrued reward receives positive utility at each round, as long as such an agent exists.

Algorithm 2 LINEARGREEDY

```
1: Initialize $u_0 = (0, \dots, 0)$
2: for $t = 1, \dots, T$ do
3:     for $i = 1, \dots, n$ do
4:         Set $0 < \delta_i < \frac{1}{nK}$
5:     end for
6:     Set $w_i = \frac{1}{\max\{\delta_i, u_{t-1}(i)\}}$ for all $i$
7:     Choose $v_t \in \text{argmax}_{v \in V_t} w \cdot v$
8:     $u_t = u_{t-1} + v_t$
9: end for
```

We now discuss some behavior of the GREEDY and LINEARGREEDY algorithms. We begin with a lemma which follows easily from the choice of ϵ_i.

Lemma 1. *For all j, k such that $1 \leq k < k + j \leq n$, and all sets of agents I and I', of size $k + j$ and k respectively,*

$$(K+1)^{n-k-j} \prod_{i \in I} \epsilon_i < \prod_{i' \in I'} \epsilon_{i'}.$$

We next state a simple interpretation of GREEDY. Let $NSW^+(v)$ be the product of all non-zero entries in v.

Proposition 1. *At every round,* GREEDY *selects an alternative to maximize the number of agents with $u_t(i) > 0$. Subject to this condition, and holding fixed the set of agents with non-zero utility,* GREEDY *chooses an alternative which maximizes $NSW^+(u_t^{avg})$.*

Indeed, we can show that every alternative not ruled out by Theorem 1 can be chosen by GREEDY, for some choice of $\{\epsilon_i\}$.

Theorem 1. *Suppose alternative a maximizes the number of agents with $u_t(i) > 0$. Suppose further that for all j such that choosing j results in the same set of agents with non-zero accrued reward, $NSW^+(u_{t-1}+V_t(\cdot, j)) \leq NSW^+(u_{t-1}+V_t(\cdot, a))$. Then a is chosen by* GREEDY *for some choice of $\{\epsilon_i\}$.*

Unlike GREEDY, LINEARGREEDY may leave some agents with zero utility even when it was possible to give positive utility to all agents.

Example 1. *Let $n = 2$, $m = 3$, and suppose that $V_1 = \left(\begin{smallmatrix} 3 & 0 & 1 \\ 0 & 3 & 1 \end{smallmatrix}\right)$. The columns represent alternatives a_1, a_2, and a_3 respectively, and the rows represent agents i_1 and i_2 respectively.*

For any choice of $0 < \epsilon_1, \epsilon_2 << 1$, GREEDY *chooses a_3 since $3\epsilon_i < 1$. However,* LINEARGREEDY *assigns the agents weights w_1, w_2 and chooses $\operatorname{argmax}_{v \in \{a_1, a_2, a_3\}} w \cdot v$. Since it must be the case that either $3w_1 > w_1 + w_2$ or that $3w_2 > w_1 + w_2$, it is not possible for a_3 to be chosen by* LINEARGREEDY *even though it is the only alternative which gives both agents positive utility.*

We can, however, provide a weaker guarantee for LINEARGREEDY.

Proposition 2. LINEARGREEDY *always chooses an alternative v_t with $v_t(i) > 0$ for at least one agent i with $u_{t-1}(i) = 0$, if such an alternative exists.*

3.1 Computational Considerations

Clearly, when the number of allocations, m, is not too large, the outcome of both GREEDY and LINEARGREEDY can be computed efficiently. However, consider a setting in which every round is a combinatorial allocation problem, so the number of alternatives is exponential in the number of items being allocated. For instance, if every round is an allocation of food bank items [6] to different charities then we will have substitutes and complements which must be taken into account, thus charities have preferences over subsets of items, not just items themselves. In this setting, computing the chosen alternative under GREEDY is weakly NP-hard even in a very restricted case.

Proposition 3. *Computing the chosen alternative v_t under* GREEDY *is weakly NP-hard, even when there are only two agents and each has additive valuations over the items.*

Note that for the LINEARGREEDY algorithm, computing the chosen alternative is equivalent to the combinatorial auction winner determination (CAWD) problem, which has been studied extensively [30, 7, 21]. Thus, the outcome under LINEARGREEDY can be computed efficiently under exactly the same conditions as the CAWD problem.[2] Even in those cases where LINEARGREEDY can not be computed efficiently, we can use any existing algorithm for the CAWD problem. This suggests that there may be significant computational advantages to using LINEARGREEDY over GREEDY as an allocation mechanism.

3.2 Axiomatization of LINEARGREEDY

It is possible to simply consider LINEARGREEDY an approximation to GREEDY. However, in this section we provide an axiomatization of the LINEARGREEDY mechanism which provides some

[2]For example, when agents have preferences over bundles of size at most 2, the problem is in P.

justification for seeing it as a worthwhile rule in and of itself, without needing to appeal to its approximation of GREEDY.

A DSCF is scale-free if it is not affected by a uniform (multiplicative) scaling of some agent's valuations. This property is desirable because it means we do not require any sort of agreement or synchronization as to the units of measurement used by the agents in their reporting.

Definition 1. *Let $c > 0$. Say that a DSCF satisfies* scale-free-ness (SF) *if the chosen alternative at round t is still among the set of (possible) chosen alternatives if we replace $v(i)$ by $c \cdot v(i)$ for all $v \in V_t$ for every $t = 1, \dots, T$.*

Lemma 2. LINEARGREEDY *satisfies SF.*

A DSCF is separable into single-minded agents if the chosen alternative at a given round is unchanged by replacing an agent by several new agents with the same accrued utility, each of which has unit positive valuation for only one alternative.

Definition 2. *Say that a DSCF is* separable into single-minded agents (SSMA) *if, at round t, the same allocation is chosen if we replace each agent with several new agents according to the following scheme: An agent with valuation vector $V_t(i, :)$ is, for each $j \in \{1, \dots, m\}$, replaced by $V_t(i, j)$ new agents, each with valuation vector e_j. Each new agent has the same (possibly hallucinated) accrued utility as the original agent it replaces.*

Lemma 3. LINEARGREEDY *satisfies SSMA.*

The plurality axiom says that if all agent valuation vectors are unit vectors, and we have no reason to distinguish between agents, then the allocation favored by the most agents should be chosen.

Definition 3. *Say that an allocation satisfies* plurality (P) *if, when all agents have preferences of the form e_j, and all agents have the same (non-zero) accrued utility, then the chosen alternative is the one with non-zero valuation from the most agents.*

The axiom says nothing about the case when all agents have zero accrued utility. The idea of the axiom is that we should choose the alternative which provides the greatest utility gain, relative to what agents already have. However, in the case that agents have zero accrued utility, it is not possible to make accurate comparisons as to the relative benefit each agent receives.

Observation 1. LINEARGREEDY *satisfies plurality.*

We now show that any mechanism that achieves SF, SSMA, and P simultaneously must agree with LINEARGREEDY, provided that all accrued rewards are non-zero.

Theorem 2. *Suppose that $u_{t-1}(i) > 0$ for all i. Denote by J_t the set of all alternatives that may be chosen by* LINEARGREEDY *at time t. If a DSCF satisfies SF, SSMA, and P then it must choose some alternative from J_t at time t.*

4 Distributional Update Algorithm

So far we have assumed nothing about the way that the input matrices are drawn. In this section, we will assume that there is some distribution, D, over $\mathbb{Z}^{n \times m}_{\geq 0, \leq K}$ from which matrices are drawn i.i.d at each round.[3]

Suppose first that we know D, and that $T = \infty$. Then the optimal solution is defined by a policy: when $V_t = v$, choose allocation j with probability x_{vj}. We simply need to choose values for $\{x_{vj}\}$

[3]In practice, this algorithm may be suitable when we believe the distribution of inputs to be somewhat stable over time.

in order to maximize $\mathbb{E}(NSW(u_t^{\text{avg}}))$, as $t \to \infty$. We can compute these variables by the following convex program:

$$\text{maximize} \sum_{i=1}^{n} \log\Big(\sum_{v \in \mathbb{Z}^{n\times m}_{\geq 0, \leq K}} \sum_{j=1}^{m} Pr(V_t = v) x_{vj} v(i,j) \Big) \tag{2}$$
$$\text{subject to} \sum_{j \in A} x_{vj} = 1 \quad \forall v \in \mathbb{Z}^{n\times m}_{\geq 0, \leq K}, \qquad x_{vj} \geq 0 \;\; \forall v, j$$

Theorem 3. *The variables x_{vj} computed by convex program 2 define the optimal policy when the distribution D is known and $T = \infty$.*

Let us now relax the assumption that D is known to the algorithm. In this case, one approach would be to approximately learn the distribution by sampling, then compute the optimal policy according to the learned distribution, and act accordingly for the remaining rounds. We can even continue to update our belief on the distribution as often as we want, re-compute the variables $\{x_{vj}\}$, and choose according to them until we perform another update step. If $T = \infty$, we can learn the distribution arbitrarily well, and behave close to optimally in the long term.

The algorithm we present now uses the same heuristic even when T is finite. We begin with no knowledge of D, but update our belief with every new piece of information V_t, and use the inferred distribution to compute a policy $\{x_{vj}\}$.

Algorithm 3 UPDATE

```
1: for $t = 1, \dots, T$ do
2:     for $s \in \{V_1, V_2, \dots, V_t\}$ do
3:         Let $p_s$ = (number of times $s$ has been realized)/$t$
4:     end for
5:     Solve Convex Program 2 using inferred probabilities $p_s$
6:     Randomly draw $v_t$ according to $x_{V_t j}$
7: end for
```

Crucially, the update to p_s is done *before* v_t is actually chosen according to $x_{V_t j}$. Were this not the case, the algorithm would not be defined when valuation matrix s appears for the first time.

Example 2. *Let $n = m = 2$. Suppose that $V_1 = \left(\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right)$, where the columns represent alternatives a_1 and a_2 respectively. Then the algorithm updates its belief on D to be that V_1 appears with probability 1, in which case the optimal polcy is to choose a_1 and a_2 with probability 0.5 each. Suppose it randomly chooses a_1. Suppose that $V_2 = \left(\begin{smallmatrix} 2 & 0 \\ 0 & 1 \end{smallmatrix}\right)$. Then the algorithm updates its belief on D to be $0.5V_1 + 0.5V_2$. Given this distribution, the optimal policy is to choose a_1 when V_2 is realized and a_2 when V_1 is realized. Thus, the algorithm chooses $v_2 = a_1$. Now suppose that $V_3 = V_2$. Then the updated belief on D is $\frac{1}{3}V_1 + \frac{2}{3}V_2$. The optimal policy now is to choose a_2 whenever V_1 is realized, and whenever V_2 is realized to draw randomly from $\frac{3}{4}a_1 + \frac{1}{4}a_2$. Thus the algorithm draws an allocation for v_3 from this distribution.*

Observe that, in Example 2, from the perspective of the algorithm at $t = 3$, a mistake was made at round 1 by choosing a_1 instead of a_2. As stated, this algorithm does nothing to take the mistake into account. However, one could imagine incorporating a more 'backwards-looking' approach into this algorithm. As a simple example we could, with probability p, simply use GREEDY at round t, which would act to partially compensate for past mistakes. In Example 2, GREEDY would choose a_2 to make up for agent 2 not accruing any utility from the first two rounds.

5 Stochastic Convex Programming Approach

Agrawal and Devanur [5] have designed algorithms for a general class of problems that encompasses our framework. In their setting, the input is a concave, Lipschitz-continuous function $f : [0,1]^n \to \mathbb{R}$, a convex set $S \subseteq [0,1]^n$, and the goal is to choose a vector v_t at each round so that $f(u_T^{avg})$ is maximized, subject to $u_T^{avg} \in S$. For the setting in our paper, however, there is no constraint, since all input vectors are feasible. That is, $S = [0,1]^n$.

They provide an algorithm using tools from convex optimization which, in our setting, reduces to Algorithm 4. The algorithm assigns a vector of weights, ϕ, to the agents and minimizes the (possibly negative) weighted sum of valuations at each round. Every round, ϕ is updated by an online convex optimization update (our implementation uses the gradient descent algorithm to update ϕ).

The initialized variables ϕ and η can be set to any values satisfying the constraints. In our implementation, we set $\phi = -\mathbf{1}$ and $\eta = 0.7$ after some experimentation.

Algorithm 4 STOCHASTIC

```
1: Initialize $\phi \in \mathbb{R}^n$, $\ell > 0$, $||\phi||_2 \leq \frac{1}{\ell}$, $\eta > 0$.
2: for $t = 1, \ldots, T$ do
3:     $V_t \leftarrow \frac{V_t}{K} + \ell$
4:     Choose $v_t = \operatorname{argmin}_{j \in V_t} j \cdot \phi_t$
5:     if $\phi_t(i) \geq \frac{-1}{\ell+1}$ then
6:         Set $\phi_{t+1}(i) = \phi_t(i) - \eta(1 - v_t(i))$
7:     else
8:         Set $\phi_{t+1}(i) = \phi_t(i) + \eta(v_t(i) + \ell + \frac{1}{\phi_t(i)})$
9:     end if
10:    if $||\phi_{t+1}||_2 > \frac{1}{\ell}$ then
11:        Set $\phi_{t+1} = \frac{1}{\ell||\phi_{t+1}||_2}\phi_{t+1}$
12:    end if
13: end for
```

Unfortunately, for our function $f = \ln NSW$ is not Lipschitz-continuous at 0. For this reason we add a constant ℓ to all utilities. On domain $[\ell, \infty)^n$, f is $\frac{1}{\ell}$-Lipschitz and therefore satisfies the required conditions.

Agrawal and Devanur prove a regret bound on Algorithm 4 of $L \cdot O\left(\sqrt{\frac{n\log(n)}{T}}\right)$, where $L = \frac{1}{\ell}$ is the Lipschitz constant for f. There are three drawbacks, however.

- The regret bound is a bound on the (concave) function $f = \ln NSW(\cdot)$, not on our actual objective, $NSW(\cdot)$.
- The regret bound is on the *expected regret* when the input matrices appear in a random order. It is *not* a guarantee on any single instance. Therefore, while we would expect good performance from this algorithm on instances in which the input matrices $\{V_t\}$ are permuted randomly, we may not necessarily expect low regret on instances where the agents' preferences change over time in a structured way.
- Since we add ℓ to all utilities, we are no longer even solving the original problem! This problem stems from the fact that the solution maximizing the Nash social welfare is not stable with respect to additive transformation. As long as ℓ is small compared to the reported utilities, we are not changing the problem too much and would still expect good solutions. However, if ℓ is large compared to the reported utilities (for example, if many reported utilities are zero), then we are making a significant adjustment to the problem which could result in bad solutions. Further, the Lipschitz constant $\frac{1}{\ell}$ appears as a linear factor in the regret bound, so

as we decrease ℓ (thus solving a problem that is closer to the actual instance), the regret bound on the solution gets worse.

We explore these issues further in Section 6.

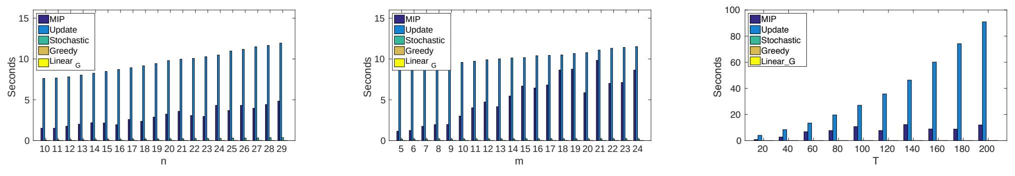
6 Experiments

6.1 Simulated Data

We compare the four algorithms discussed in this paper – GREEDY, LINEARGREEDY, STOCHASTIC, and UPDATE– on input data randomly generated from a variety of distributions. As a benchmark we also compute the optimal offline solution for each input using MIP (1).

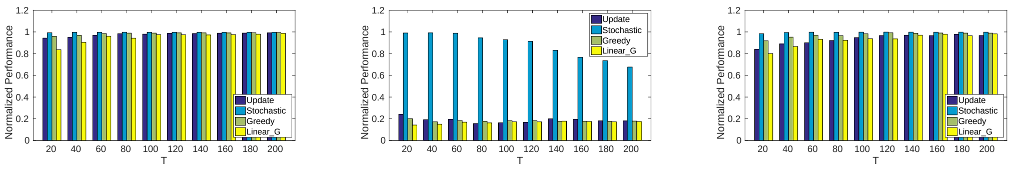
We consider three input models. The first, *uniform*, has each $V_t(i,j)$ chosen uniformly at random between 0 and 20. The second, *half-half*, draws V_t from a different distribution depending on t. For $t < \frac{T}{2}$, $V_t = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$, where A, B, C, D are submatrices of size $\frac{n}{2} \times \frac{m}{2}$. Entries in A are integers in the range 0 to 25 drawn uniformly at random, entries in B and C are in the range 0 to 5, and entries in D are in the range 0 to 10. For $t \geq \frac{T}{2}$, submatrices A, B, C, D are drawn in the same way but $V_t = \left(\begin{smallmatrix} D & B \\ C & A \end{smallmatrix}\right)$. The third model, *alternating*, sets $V_t = \left(\begin{smallmatrix} A & B \\ C & D \end{smallmatrix}\right)$ for odd t and $V_t = \left(\begin{smallmatrix} D & B \\ C & A \end{smallmatrix}\right)$ for even t. In both of these latter models, it is almost always optimal to choose an alternative for which half of the valuations are being drawn from the high, 'A', distribution. The other agents can be compensated in a round where they draw from the 'A' distribution.

For every fixed value of n, m, T, and input model that we report, values are averaged over 15 random instances. When not explicitly varied, $n = 20$, $m = 10$, and $T = 40$.



(a) Running time, n varies (b) Running time, m varies (c) Running time, T varies

Figure 1: Simulation results showing the effect of varying number of agents, n, number of alternatives, m, and number of rounds, T, on the runtime of each algorithm.



(a) Value, uniform model (b) Value, half-half model (c) Value, alternating model

Figure 2: Simulation results showing the effect of the input model on the value achieved by the algorithms. For each model, n and m are held constant while T varies.

Consider first the runtime comparisons in Figure 1. These simulations are performed on inputs drawn from the half-half model, varying values of n, m, and T seperately. Three of the algorithms, GREEDY, LINEARGREEDY, and STOCHASTIC, take virtually no time to run on the instances we consider. This is not surprising as each makes only a simple comparison between each of the m

Table 1: Spark Workloads

App	Category	Dataset	Data Size
`Correlation`	Statistics	kdda2010 [31]	2.5G
`DecisionTree`	Classification	kdda2010	2.5G
`FP Growth`	Pattern Mining	Webdocs [22]	1.5G
`GradientBoostedTrees`	Classification	kddb2010 [31]	4.8G
`KMeans`	Clustering	uscensus1990 [3]	327M
`LinearRegression`	Classification	kddb2010	4.8G
`ALS`	Collaborative Filtering	movielens2015 [2]	325M
`NaiveBayesian`	Classification	kdda2010 [31]	2.5G
`SVM`	Classification	kdda2010	2.5G
`Pagerank`	Graph Processing	wdc2012 [4]	5.3G
`ConnectedComponents`	Graph Processing	wdc2012	5.3G
`TriangleCounting`	Graph Processing	wdc2012	5.3G

alternatives, followed by some very simple arithmetic operations to update weights and accrued utility. The UPDATE algorithm is the slowest by far on our simulations, even slower than the MIP for solving the offline problem (although we would expect that for large values of T, the MIP would become slower than UPDATE). We could speed it up by a constant factor of k by only updating the inferred distribution, and values of x_{V_tj}, every k rounds, and still expect reasonable results. All of our algorithms scale well with n and m. Runtime results for the other two input models are very similar and we do not present them here.

Turning to the value comparisons in Figure 2, we see that the input model used heavily influences the performance of the algorithms. In these graphs, OPT is normalized to 1, and for each input model we present results only for varying T. The results for varying m and n look very similar.

For both the uniform and alternating input models, all algorithms perform well, achieving at least 80% of the optimal value. These models provide relatively simple cases for the algorithms; indeed, simply maximizing (additive) welfare at each round is the optimal solution in the limit as T grows. However, for the half-half distribution STOCHASTIC is clearly better than all the other algorithms, achieving over 60% of OPT compared to less than 30% for the others. Interestingly, the performance of STOCHASTIC is quite clearly *decreasing* as T increases. This is because, in these instances, the optimal offline solution simply keeps giving more and more utility to the well-off agents until the halfway point, and then is able to change once the other agents start getting high valuations. Of course, as T increases, any algorithm must starve certain agents for increasingly long periods of time to be competitive with OPT. That STOCHASTIC is competitive for the smaller values of T simply reflects that it is willing to be patient and starve certain agents for some number of rounds, but eventually starts giving them utility at the expense of efficiency.

6.2 Real Data: Power Boost Allocation

We ran the algorithms on real data gathered from a power boost allocation problem. In this problem, n computer applications are each allocated a base level of power, and compete for $m < n$ additional (indivisible) units of extra power (*power boosts*) at each of T rounds. For our instance, power boosts are allcoated using RAPL [1] technology and each application's performance is measured under base and high power limits, 30W and 130W, respectively. We evaluate Apache Spark [33] benchmarks. Table 1 lists the twelve Spark applications in our instance.

Each Spark application is defined by a fixed number of tasks. We profile tasks' completion time. We define an application's utility in a round as the number of tasks completed normalized by its total number of tasks. Since the length of the utility trace is shorter when profiled under boosted power, we use linear interpolation to extend the shorter trace. Thus, for each application a, we estimate the base power utility ($u_{a,t}^{\text{base}}$) and boosted power utility ($u_{a,t}^{\text{boost}}$) in each round.

In our instance, there are two power boosts to be allocated. Therefore, at each round there are $\binom{12}{2}$ alternatives, one for each pair of applications. For an alternative j corresponding to power boosts for applications a and b, we have that $V_t(a,j) = u_{a,t}^{\text{boost}}$, $V_t(b,j) = u_{b,t}^{\text{boost}}$, and $V_t(c,j) = u_{c,t}^{\text{base}}$

for all other applications $c \neq a, b$. We have 120 rounds in the instance we tested.

The Nash Social Welfare achieved on this instance is shown in Figure 3, normalized against OPT. Most striking is the poor performance of STOCHASTIC. We hypothesize that this is due to some of the applications having long stretches of consecutive rounds where they achieve zero utility for all allocations, followed by short periods with positive reported utility. Of the issues that we discussed regarding the regret guarantees of the STOCHASTIC algorithm in Section 5, both (1) and (3) are relevant here, due to the high number of zeros present in the data. We have not yet conducted further experiments to determine the extent to which each of these issues individually affects the performance of STOCHASTIC on this instance, and whether STOCHASTIC performs poorly on other real-world instances.

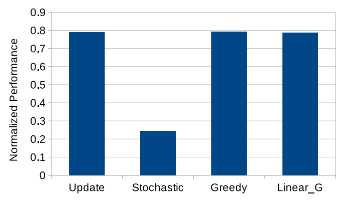


Figure 3: Nash Social Welfare achieved by the algorithms, as a fraction of OPT.

Runtime results are similar to those presented in Section 6.1, with one exception. The time taken to solve the offline instance is 384 seconds, whereas UPDATE takes only 108 seconds. This provides evidence that, for large instances, the optimal MICP is prohibitively slow compared to our online algorithms. For comparison, the three other algorithms each ran in less than 0.2 seconds.

7 Applications

7.1 Voting

Our setup can be directly applied to voting, where agents are voters who report a utility for each of the alternatives. If voters only report an ordering over alternatives within each round, then we can simply infer utilities according to a chosen *scoring vector*. For example, we could set $V(i, j) = 1$ if alternative j is voter i's most preferred alternative and 0 otherwise (the *plurality* scoring vector), or $V(i, j) = m - k$ when j is ranked k-th in i's preference order (the *Borda* utility vector).

An interesting direction for future work is to investigate what social-choice theoretic properties are satisfied by the Nash social welfare in this repeated setting. One weak property is *unanimity*, which states that if all voters rank the same alternative at the top of their ordering, then that alternative should be chosen (in the dynamic setting we could require this on a round-by-round basis). Clearly all of the algorithms presented in this paper satsify this condition for all monotone scoring vectors. Some other fundamental axioms also extend naturally to the dynamic setting, for example *anonymity* and *neutrality* (which we define for the dynamic setting in Section 2).

For some axioms, however, it is not so clear how to extend to the dynamic setting. For instance, consider the *Condorcet criterion*, which states that any alternative which achieves a pairwise majority against all other alternatives should be chosen. This makes sense in the one-shot setting, but maybe less sense in the dynamic case. Suppose that there are two alternatives, A and B, and that 51% of voters prefer A to B in every round. Then the Condorcet criterion appears to say that we should choose A in every round, while fairness considerations dictate choosing B at least occasionally. It is not clear how we would extend the Condorcet criterion to the dynamic setting and, if we cannot, we may need novel axioms.

There is a natural link between repeated elections and the theory of multi-winner elections. In multi-winner elections, not only do we want to choose popular alternatives, but we also want to represent as many voters as possible, for which several rules have been designed [12, 23, 20]. Consider an election where the aim is to choose a committee of size $k < m$. This is exactly equivalent to setting $T = k$ and choosing a single distinct winner at each round, while also imposing the restriction that voters do not change their votes between rounds. Thus we can view multi-winner elections as a special case of repeated elections. It would be interesting to check whether any desiderata in the context of multi-winner elections extend naturally to the repeated setting.

7.2 Allocating Shared Resources

Consider a situation in which a group of agents take turns being allocated a shared resource for discrete units of time. Examples include allocating supercomputer time among members of a university department or assigning the use of a holiday home owned jointly by several people. In both cases, demand varies across time intervals and across agents. For instance, people who like skiing may want use of the holiday home in winter, while those who like hiking may prefer a different season.

Another interesting aspect of these situations is that our notion of fairness may not be to treat all agents exactly equally. For instance, if people contributed unequally to the purchase of the holiday home, the group may decide that someone who paid twice as much as another person 'deserves' to get twice the benefit from the home. In the supercomputer example, we may wish to allocate time based on the amount of grant money contributed to the purchase of the machine (for example).

In these cases we may wish to generalize the Nash social welfare to the *Cobb-Douglas welfare*. The Cobb-Douglas welfare for utility vector v, $CD(v)$, is given by $CD(v) = \prod_{i=1}^{n} v(i)^{\alpha_i}$, where $\sum_{i=1}^{n} \alpha_i \leq 1$. The case where all $\alpha_i = \frac{1}{n}$ is the special case of Nash social welfare, but setting other values of α_i allows us to prioritize some agents over others. It is illuminating to consider the simple case where all agents have a common unit of utility (say, dollars). In this case, the Nash social welfare is maximized when all agents receive exactly the same utility. If we generalize the coefficients, then the Cobb-Douglas welfare is maximized when the agents receive utility in exactly the ratio of their exponents α_i. So if agent i contributed twice as much money to the purchase of the holiday home as agent j, simply set $\alpha_i = 2\alpha_j$.

8 Conclusion

Election designers and social choice researchers often do not consider the fact that many elections are conducted as sequences of related elections. In this work, we have provided a framework to allow for the design and analysis of dynamic election protocols, and repeated decision making rules generally. We have presented four candidate online algorithms for solving these dynamic problems. Our simulations do not determine a clear winner, but suggest that the right choice of algorithm is highly dependent on the setting and the model of how agents' valuations change over time.

Our work is preliminary, and leaves a lot of scope for future research in addition to the specific directions already discussed. One direction would be to design a more precise model of voter preferences, possibly modeling changing preferences by an MDP as has been done in [9, 27]. We have also not considered modeling discounting of the agents' utilities. Additionally, we have treated the Nash social welfare as a substitute for any explicit fairness criterion (for example, envy-freeness). While it does satisfy some desirable properties, it is an interesting topic for future work to consider other ways of arriving at a 'fair' solution.

Finally, there are many interesting questions regarding strategic behavior by the agents. In the most general setting, there appears to be no hope for a fair, strategy-proof rule due to the free-rider problem: agents are incentivized to under-report their utility for an alternative that gets chosen, and are thus indistinguishable from an agent that is genuinely unhappy with the chosen alternative. However, it may be possible to regain some (limited) strategy-proofness in a more restricted setting. For instance, what if we place restrictions on the utilities that can be reported, or restrict our attention to private goods?

Acknowledgments

We sincerely thank Songchun Fan for sharing the data used in Section 6.2. We are thankful for support from NSF under awards IIS-1527434, IIS-0953756, CCF-1101659, CCF-1149252 (CA- REER), CCF-1337215 (XPS-CLCCA), SHF-1527610, and AF-1408784, ARO under grants W911NF-12-1-0550 and W911NF-11-1-0332, and a Guggenheim Fellowship. This work is also supported by STARnet, a Semiconductor Research Corporation program, sponsored by MARCO and DARPA. This work was done in part while Conitzer was visiting the Simons Institute for the Theory of Computing. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of these sponsors.

References

[1] Intel 64 and ia-32 architectures softawre developer's manual. `https://www-ssl.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-vol-3b-part-2-manual.pdf`.

[2] Movielens. `http://grouplens.org/datasets/movielens/`.

[3] Us census data (1990) data set. `https://archive.ics.uci.edu/ml/datasets/US+Census+Data+(1990)`.

[4] Web data commons: Hyperlink graphs. `http://webdatacommons.org/hyperlinkgraph/index.html`.

[5] S. Agrawal and N. R. Devanur. Fast algorithms for online stochastic convex programming. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1405–1424. SIAM, 2015.

[6] M. Aleksandrov, H. Aziz, S. Gaspers, and T. Walsh. Online fair division: analysing a food bank problem. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI-15)*, 2015.

[7] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multiagent Systems*, pages 39–46. IEEE, 2000.

[8] H. Aziz, M. Brill, V. Conitzer, E. Elkind, R. Freeman, and T. Walsh. Justified representation in approval-based committee voting. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 784–790, Austin, TX, USA, 2015.

[9] C. Boutilier and A. D. Procaccia. A dynamic rationalization of distance rationalizability. In *AAAI*, 2012.

[10] I. Caragiannis, J. A. Covey, M. Feldman, C. M. Homan, C. Kaklamanis, N. Karanikolas, A. D. Procaccia, and J. S. Rosenschein. On the approximability of dodgson and young elections. In *Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1058–1067. Society for Industrial and Applied Mathematics, 2009.

[11] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and A. D. Procaccia. Socially desirable approximations for dodgson?s voting rule. *ACM Transactions on Algorithms (TALG)*, 10(2):6, 2014.

[12] J. R. Chamberlin and P. N. Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review*, 77(03):718–733, 1983.

[13] R. Cole and V. Gkatzelis. Approximating the Nash social welfare with indivisible items. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing*, pages 371–380. ACM, 2015.

[14] A. Darmann and J. Schauer. Maximizing Nash product social welfare in allocating indivisible goods. *European Journal of Operational Research*, 247(2):548–559, 2015.

[15] E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 53–60. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[16] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica. Dominant resource fairness: Fair allocation of multiple resource types. *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, pages 24–24, 2011.

[17] M. Guo, V. Conitzer, and D. M. Reeves. Competitive repeated allocation without payments. In *Proceedings of the Fifth Workshop on Internet and Network Economics (WINE)*, pages 244–255, Rome, Italy, 2009.

[18] A. Jalali, R. Padovani, and R. Pankaj. Data throughput of cdma-hdr a high efficiency-high data rate personal communication wireless system. In *Vehicular technology conference proceedings, 2000. VTC 2000-Spring Tokyo. 2000 IEEE 51st*, volume 3, pages 1854–1858. IEEE, 2000.

[19] I. Kash, A. D. Procaccia, and N. Shah. No agent left behind: Dynamic fair division of multiple resources. *Journal of Artificial Intelligence Research*, pages 579–603, 2014.

[20] D. M. Kilgour. Approval balloting for multi-winner elections. In *Handbook on approval voting*, pages 105–124. Springer, 2010.

[21] D. Lehmann, R. Müller, and T. Sandholm. The winner determination problem. *Combinatorial auctions*, pages 297–318, 2006.

[22] C. Lucchese, S. Orlando, R. Perego, and F. Silvestri. WebDocs: a real-life huge transactional dataset. In *Workshop on Frequent Itemset Mining Implementations*, 2004.

[23] B. L. Monroe. Fully proportional representation. *American Political Science Review*, 89(04):925–940, 1995.

[24] J. F. Nash Jr. The bargaining problem. *Econometrica: Journal of the Econometric Society*, pages 155–162, 1950.

[25] D. Parkes, A. Procaccia, and N. Shah. Beyond dominant resource fairness: Extensions, limitations, and indivisibilities. In *ACM Conference on Electronic Commerce*, pages 808–825. ACM, 2012.

[26] D. C. Parkes, R. Cavallo, F. Constantin, and S. Singh. Dynamic incentive mechanisms. *AI Magazine*, 31(4):79–94, 2010.

[27] D. C. Parkes and A. D. Procaccia. Dynamic social choice with evolving preferences. AAAI, 2013.

[28] A. D. Procaccia. Cake cutting: not just child's play. *Communications of the ACM*, 56(7):78–87, 2013.

[29] S. Ramezani and U. Endriss. *Nash social welfare in multiagent resource allocation*. Springer, 2010.

[30] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial intelligence*, 135(1):1–54, 2002.

[31] J. Stamper, A. Niculescu-Mizil, S. Ritter, G. Gordon, and K. Koedinger. Algebra i 2006-2007. challenge data set from kdd cup 2010 educational data mining challenge. `http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp`.

[32] P. Viswanath, D. N. Tse, and R. Laroia. Opportunistic beamforming using dumb antennas. *Information Theory, IEEE Transactions on*, 48(6):1277–1294, 2002.

[33] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing*, volume 10, page 10, 2010.

Rupert Freeman
Department of Computer Science
Duke University
Durham, NC, USA
Email: `rupert@cs.duke.edu`

Seyed Majid Zahedi
Department of Computer Science
Duke University
Durham, NC, USA
Email: `zahedi@cs.duke.edu`

Vincent Conitzer
Department of Computer Science
Duke University
Durham, NC, USA
Email: `conitzer@cs.duke.edu`

A Proof of Lemma 1

Lemma 1. *For all j, k such that $1 \leq k < k + j \leq n$, and all sets of agents I and I', of size $k + j$ and k respectively,*

$$(K+1)^{n-k-j} \prod_{i \in I} \epsilon_i < \prod_{i' \in I'} \epsilon_{i'}.$$

Proof. Let $1 \leq k < k + j \leq n$. Let $I = \{i_1, \ldots, i_{k+j}\}$ and $I' = \{i'_1, \ldots, i'_k\}$. Recall that

$0 < x < \frac{1}{2^n(K+1)^{n(n+1)}}$ and $x \le \epsilon_i \le 2(K+1)^{n-1}x$ for all i. Then

$$\begin{aligned}
(K+1)^{n-k-j}\prod_{i\in I}\epsilon_i &\le (K+1)^{n-k-j}(2(K+1)^{n-1}x)^{k+j} \\
&< x^k(K+1)^{n-k-j}(2(K+1)^{n-1})^{k+j}\left(\frac{1}{2^n(K+1)^{n(n+1)}}\right)^j \\
&\le x^k\frac{2^n(K+1)^{(k+j+1)n-2k-2j}}{(2^n(K+1)^{n(n+1)})^j} \\
&\le x^k\frac{2^n(K+1)^{(k+j+1)n-2k-2j}}{2^n(K+1)^{n(n+1)}} \\
&< x^k\frac{(K+1)^{n(k+j+1)}}{(K+1)^{n(n+1)}} \\
&\le x^k \\
&\le \prod_{i'\in I'}\epsilon_{i'}
\end{aligned}$$

□

B Proof of Proposition 1

Proposition 1. *At every round,* GREEDY *selects an alternative to maximize the number of agents with $u_t(i) > 0$. Subject to this condition, and holding fixed the set of agents with non-zero utility,* GREEDY *chooses an alternative which maximizes $NSW^+(v_t^{ave})$.*

Proof. Consider the action of GREEDY at round t. Suppose first that all agents have $u_{t-1}(i) > 0$. Then the first condition in the proposition statement is vacuous; all choices maximize the number of agents with $u_t(i) > 0$. Since $\epsilon_i < 1$ for all i, the choice at Line 6 is precisely to maximize $NSW(v_t^{ave}) = NSW^+(v_t^{ave})$.

Now suppose that $u_{t-1}(i) = 0$ for some agent i, and consider two alternatives v' and v. Suppose that $|I' = \{i : u_{t-1}(i) + v'(i) = 0\}| = k$ and $|I = \{i : u_{t-1}(i) + v(i) = 0\}| = k + j > k$. To show that GREEDY maximizes the number of agents with $u_t(i) > 0$, it suffices to show that the product in Line 6 is larger when v' is chosen than when v is chosen.

We consider the case in which the produc tin Line 6 is greatest for v (compared to v'). In particular, all $n - k$ agents with $u_{t-1}(i) + v'(i) > 0$ have $u_{t-1}(i) + v'(i) = 1$, and all $n - k - j$ agents with $u_{t-1}(i) + v(i) > 0$ have $u_{t-1}(i) + v(i) = K + 1$ (if $u_{t-1}(i) + v(i) > K + 1$ then $u_{t-1}(i)+v'(i) \ge u_{t-1}(i) > 1$, since no single-round valuation is greater than K). Then the product on Line 6 for v' is $\prod_{i'\in I'}\epsilon_{i'}$, and for v is $(K+1)^{n-k-j}\prod_{i\in I}\epsilon_i$. By Lemma 1, the former is greater than the latter, and GREEDY chooses v', thus minimizing the number of agents with $u_t(i) = 0$.

Finally, suppose that GREEDY chooses v_t when there exists another alternative v which results in the same set of agents with $u_t(i) > 0$ (call this set of agents I), and $NSW^+(u_{t-1} + v_t) < NSW^+(u_{t-1} + v)$. Then, by definition of NSW^+,

$$\begin{aligned}
&\prod_{i\in I}(u_{t-1}(i) + v_t(i)) < \prod_{i\in I}(u_{t-1}(i) + v(i)) \\
\iff &\prod_{i\in I}(u_{t-1}(i) + v_t(i))\prod_{i\notin I}\epsilon_i < \prod_{i\in I}(u_{t-1}(i) + v(i))\prod_{i\notin I}\epsilon_i \\
\iff &\prod_{i=1}^{n}\max\{(u_{t-1}(i) + v_t(i)), \epsilon_i\} < \prod_{i=1}^{n}\max\{(u_{t-1}(i) + v(i)), \epsilon_i\},
\end{aligned}$$

which contradicts the choice of v_t. $\square$

C Proof of Theorem 1

Theorem 1. *Suppose alternative a maximizes the number of agents with $u_t(i) > 0$. Suppose further that for all j such that choosing j results in the same set of agents with non-zero accrued reward, $NSW^+(u_{t-1}+V_t(\cdot, j)) \leq NSW^+(u_{t-1}+V_t(\cdot, a))$. Then a is chosen by* GREEDY *for some choice of $\{\epsilon_i\}$.*

Proof. Let $a \in V_t$ satisfy the conditions of the theorem statement. We exhibit a set of $\{\epsilon_i\}$ such that a is chosen by GREEDY. Let I be the set of agents with non-zero accrued reward after choosing a. Then let $\epsilon_i = x$ (the same x as in Line 4 of Algorithm 1) for all $i \in I$, and $\epsilon_{i'} = 2(K+1)^{n-1}x$ for all $i' \notin I$. Consider some alternative j that also maximizes the number of agents with non-zero accrued reward (since otherwise, by Proposition 1, it would certainly not be chosen by GREEDY), and denote by J the set of agents given non-zero accrued utility after choosing j.

Suppose that $J \neq I$. By assumption, $|J| = |I|$. Therefore $|I \backslash J| = |J \backslash I|$. Let us consider the contribution of all agents to the product in Line 6 of Algorithm 1 for alternatives j and a. There are four types of agent to consider:

1. An agent $i \in I \backslash J$. These are agents for which $u_{t-1}(i) = 0$, with $V_t(i, a) \geq 1$ and $V_t(i, j) = 0$. Therefore each of these agents contributes a factor of at least 1 to the product for a and $\epsilon_i = x$ to the product for j.

2. An agent $i \in I \cap J$. These are agents for which the choice of either a or j both result in $u_t(i) > 0$. In the worst case for a (relative to j), $u_{t-1}(i) = 1$ with $V_t(i, a) = 0$ and $V_t(i, j) = K$. That is, each of these agents contributes a factor of at least 1 to the product for a and at most $K + 1$ to the product for j.

3. An agent $i \in J \backslash I$. These are agents for which $u_{t-1}(i) = 0$, with $V_t(i, a) = 0$ and $V_t(i, j) \geq 1$. Each of these agents contributes a factor of $\epsilon_i = 2(K+1)^{n-1}x$ to the product for a and at most K to the product for j.

4. An agent $i \notin I \cup J$. These are agents for which the choice of either a or j both result in $u_t(i) = 0$. Thus they contribute exactly the same factor to the product for both a and j.

Let us write down the product from Line 6 for a (not counting the agents of type 4 which make the same contribution to both). It is at least:

$$(2(K+1)^{n-1}x)^{|J\backslash I|} = (2(K+1)^{n-1}x)^{|I\backslash J|} \tag{3}$$

The product for j (not counting the agents of type 4 which make the same contribution to both) is at most:

$$\begin{aligned} x^{|I\backslash J|} \cdot (K+1)^{|I\cap J|} \cdot K^{|J\backslash I|} &\leq x^{|I\backslash J|} \cdot (K+1)^{|I\cap J|} \cdot (K+1)^{|J\backslash I|} \\ &= x^{|I\backslash J|} \cdot (K+1)^{|I\cap J|} \cdot (K+1)^{|I\backslash J|} \end{aligned} \tag{4}$$

Noting that $|I \cap J| + |I \backslash J| \leq n - 1$ (since $|I \cap J| + 2|I \backslash J| \leq n$ and $I \backslash J$ is nonempty), it is clear that expression 3 is greater than expression 4. Therefore the product in Line 6 is higher for a than for j.

Finally, suppose that $J = I$. Then, by the condition of the theorem, $NSW^+(u_{t-1} + j) \leq NSW^+(u_{t-1} + a)$, which implies that choosing a results in a weakly higher product in Line 6 than choosing j.

Therefore a is chosen by GREEDY, since all other alternatives which maximize the number of agents with non-zero accrued reward after round t have been ruled out, given our particular choice of ϵ. $\square$

D Proof of Proposition 2

Proposition 2. LINEARGREEDY *always chooses an alternative v_t with $v_t(i) > 0$ for at least one agent i with $u_{t-1}(i) = 0$, if such an alternative exists.*

Proof. Let i be an agent with $u_{t-1}(i) = 0$ and v be an alternative with $v(i) \geq 1$. Then the weight assigned to i at round t by LINEARGREEDY is $w_i = \frac{1}{\delta_i} \geq nK$, and the dot product in Line 7 of Algorithm 2 is at least nK. Suppose for contradiction that LINEARGREEDY chooses an alternative v' such that $v'(i) > 0$ only for agents with $u_{t-1}(i) > 0$. The weight of each such agent is $\frac{1}{u_{t-1}(i)} \leq \frac{1}{1} = 1$, and $v'(i) \leq K$, thus the dot product in Line 7 is at most $(n-1)K < nK$. Thus v' is not chosen by LINEARGREEDY. □

E Proof of Proposition 3

Proposition 3. *Computing the chosen alternative v_t under* GREEDY *is weakly NP-hard, even when there are only two agents and each has additive valuations over the items.*

Proof. Suppose there are two agents with the same valuation over the items in the first round. Then the allocation under GREEDY is to allocate each agent an equal share of the items according to their common valuation (or as close to equal as possible). This is exactly an instance of PARTITION, which is weakly NP-hard. □

F Proof of Lemma 2

Lemma 2. LINEARGREEDY *satisfies SF.*

Proof. Suppose that agent i scales all their valuations by $c > 0$. We show by induction that there exists some choice of δ such that LINEARGREEDY still chooses the same alternative at each round as it did before the scaling.

Consider the first round, $t = 1$. Let δ' denote the vector of hallucinated utilities chosen before the scaling. When i scales their valuations by factor c, simply set $\delta_i = c \cdot \delta'_i$,[4] and $delta_j = delta'_j$ for all $j \neq i$. Thus the weight of i is scaled by $\frac{1}{c}$, so the value of $w \cdot v$ is unchanged for all $v \in V_t$, and the same alternative is chosen at round 1.

Now consider round $t > 1$, and suppose that the same alternatives are chosen for all rounds before t. In particular, the accrued utility of all agents $i' \neq i$ (or, in the case that i' has zero accrued utility, then the value of δ_i) is the same as before the scaling, so their weights have not changed. But the accrued utility of i has scaled by a factor of c, since i's valuation for every alternative at every round is scaled by c (and, in the case that i has zero accrued utility, we still have that $\delta_i = c \cdot \delta'_i$), so that the weight of i is scaled by $\frac{1}{c}$. Thus, again, $w \cdot v$ is unchanged for all $v \in V_t$ and the same alternative is chosen.

Finally we need to rule out the possibility of there being some setting of δ such that some new alternative, a, is chosen at round t as a result of the scaling that was not previously chosen. But if this were the case, then we can just scale the scaled instance by $\frac{1}{c}$ and return to the original instance where, by the above proof, there is some value of δ such that a is chosen at round t. □

G Proof of Lemma 3

Lemma 3. LINEARGREEDY *satisfies SSMA.*

[4] It is possible that δ_i is now greater than $\frac{1}{nK}$. If this is the case, we can simply divide δ by some small constant, bringing δ_i back into the allowed range and not changing the relative weights of the agents.

Proof. Consider round t with valuation matrix V_t. LINEARGREEDY chooses

$$\operatorname*{argmax}_{j \in V_t} \sum_{i=1}^{n} \left(V_t(i,j) \frac{1}{u_{t-1}(i)} \right)$$

Now suppose that we replace the agents with new agents according to the definition of SSMA. For every agent i and alternative j, we now have $V_t(i,j)$ agents with accrued utility $u_{t-1}(i)$ and valuation vector e_j. The accrued utility, and therefore the weight of each of these agents in the LINEARGREEDY algorithm, is the same as the agent it replaced. Thus the value of the dot product on Line 7 for an alternative j is

$$\sum_{i=1}^{n} \left(V_t(i,j) \frac{1}{u_{t-1}(i)} \right),$$

and LINEARGREEDY chooses the same alternative in each case. □

H Proof of Theorem 2

Theorem 2. *Suppose that $u_{t-1}(i) > 0$ for all i. Denote by J_t the set of all alternatives that may be chosen by* LINEARGREEDY *at time t. If a DSCF satisfies SF, SSMA, and P then it must choose some alternative from J_t at time t.*

Proof. We have already shown that LINEARGREEDY satisfies SF, SSMA, and P.

Suppose that all agents have $u_{t-1}(i) > 0$. Let M be a DSCF that satisfies all three axioms simultaneously. We show that M's choice of alternative is the same as one chosen by LINEARGREEDY.

Without loss of generality, let $u_{t-1}(i) = u$ for all agents i. We may assume this because, by SF, M would choose the same allocation at round t (and all previous rounds) even if the valuation vectors of some agent(s) were multiplied by a constant across all rounds. So, if $u_{t-1}(i) \neq u_{t-1}(j)$, we can transform the instance to one in which all agents have the same accrued reward by multiplying agent i's valuations by $\prod_{j \neq i} u_{t-1}(j)$ for all i. Then all agents have the same acccrued utility, $\prod_i u_{t-1}(i)$.

By SSMA, we can replace the agents with $\sum_{j=1}^{m} V_t(i,j)$ agents, such that $V_t(i,j)$ of them have valuation vector e_j for all $j \in \{1, \ldots, m\}$, all with accrued reward u. Then, by plurality, the chosen allocation is

$$\operatorname*{argmax}_{j \in V_t} \sum_{i=1}^{n} V_t(i,j). \tag{5}$$

But note that LINEARGREEDY assigns equal weight w_i to each agent since $u_{t-1}(i) = u_{t-1}(j)$ for all i, j. Thus LINEARGREEDY chooses precisely the alternatives which maximize Equation 5. □

I Proof of Theorem 3

Theorem 3. *The variables x_{vj} computed by convex program 2 define the optimal policy when the distribution D is known and $T = \infty$.*

Proof. Let $\{x_{vj}\}$ be the optimal solution to convex program 2. We show that $\{x_{vj}\}$ converges to the optimal solution for the offline problem in the case that $T \to \infty$. So consider an offline instance for some large T, large enough that every matrix v that occurs with non-zero probability in D appears a large number of times in the input. For every round that $V_t = v$, choose an alternative by sampling from the distribution x_{vj}. Denote the objective value achieved by S_D. Now consider the observed distribution in the finite instance, O. Consider solving convex program 2 for distribution O, giving variables x_{vj}^{O}, and denote the value of the resulting solution on the large instance, S_O. It can be shown that $S_O \to S_D$ as $O \to D$. By the law of large numbers, $\lim_{T\to\infty} O = D$, therefore $\lim_{T\to\infty} S_O = S_D$.

Now consider the optimal offline solution as defined by mixed integer program 1. Denote the value it achieves by S_{MIP}. Clearly $S_{MIP} \geq S_O$, since S_{MIP} is optimal. Next, consider the alternatives chosen in the offline solution and use them to define variables

$$x'_{vj} = \#\text{times}(V_t = v \text{ and } j \text{ is chosen})/\#\text{times}(V_t = v). \tag{6}$$

Denote the value of the solution defined by the x'_{vj} variables by S'. $S' = S_{MIP}$ since sampling from the solution corresponding to S' gives the solution corresponding to S_{MIP}.

Lastly, we show that $\lim_{T\to\infty} S' \leq \lim_{T\to\infty} S_O$. Note that the variables x'_{vj} are a feasible solution to convex program 2. Therefore, as long as the instance is large enough that the probabilities x^O_{vj} can be well-sampled for every v that appears in the instance, S_O is the highest value that can be achieved. As $T \to \infty$, we can sample these variables arbitrarily well. Thus, $\lim_{T\to\infty} S' \leq \lim_{T\to\infty} S_O$.

So, in the limit as $t \to \infty$, we have the relations

$$S_{MIP} = S' \leq S_O = S_D \leq S_{MIP}. \tag{7}$$

Here the inequalities are forced to be equalities, otherwise we get $S_{MIP} < S_{MIP}$. In particular, $\lim_{T\to\infty} S_D = S_{MIP}$. $\square$